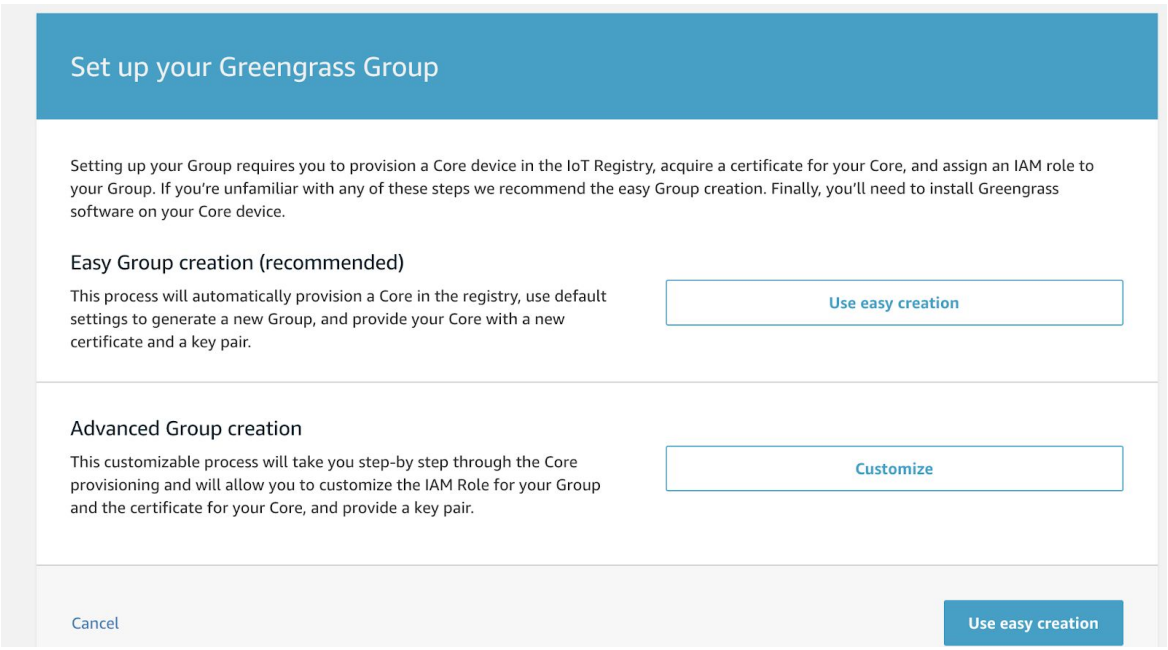
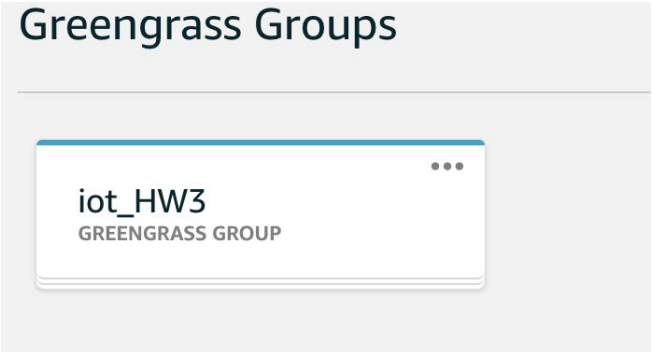
CSS532 HW3 Report
Name: Iris(Qiaoyu) Zheng

1 Feature Realization：

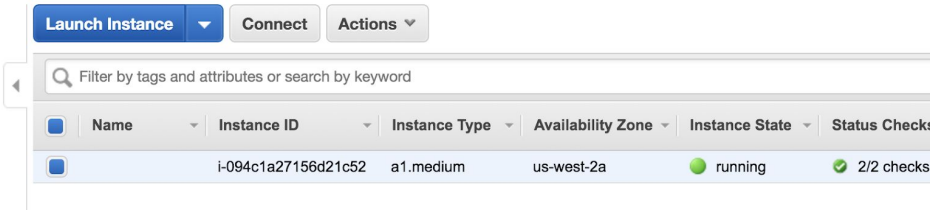
a).Create an AWS Greengrass group



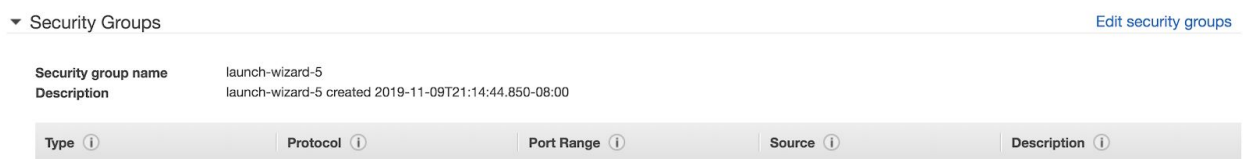


Name the group as iot_HW3

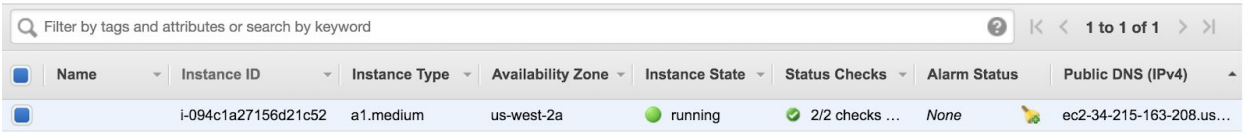
b). Set up an EC2 micro instance as an AWS Greengrass Core



Click launch Instance on EC2 page, customize the system as Ubutun 64-bit(Arm), for future utilization with raspberry pi.



Do customization, especially, add MQTT protocol into the security group.



Successfully launch the ex2 instance

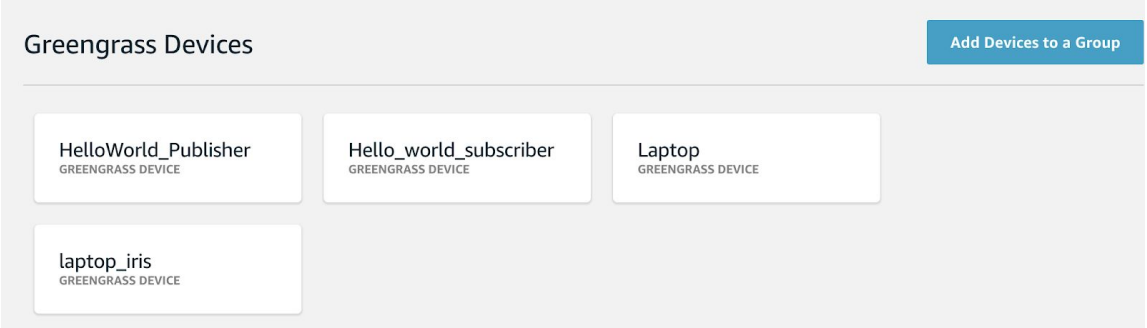
Use the certification given when registering the instance to log into the ec2 instance, use the certification given when registering the core to install the Greengrass core software package. Install the AWS Greengrass Core SDK on this EC2 instance terminal and run the Daemon

```
ubuntu@ip-172-31-53-203:/greengrass$ cd certs/
ubuntu@ip-172-31-53-203:/greengrass/certs$ ls
579ffac3ad.cert.pem  579ffac3ad.private.key  579ffac3ad.public.key  README  root.ca.pem
ubuntu@ip-172-31-53-203:/greengrass/certs$ vim 579ffac3ad.public.key
ubuntu@ip-172-31-53-203:/greengrass/certs$ cd /greengrass/ggc/core/
ubuntu@ip-172-31-53-203:/greengrass/ggc/core$ sudo ./greengrassd start
Setting up greengrass daemon
Validating hardlink/softlink protection
Waiting for up to 1m10s for Daemon to start

Greengrass successfully started with PID: 3485
ubuntu@ip-172-31-53-203:/greengrass/ggc/core$ ps aux | grep 3485
root      3485  0.3  0.7 463400 15092 pts/0    Sl   08:16   0:00 /greengrass/ggc/packages/1.9.4/bin/daemon -core-dir /greengrass/ggc
/packages/1.9.4 -greengrassdPid 3482
ubuntu    3504  0.0  0.0   6100   644 pts/0    S+   08:17   0:00 grep --color=auto 3485
```

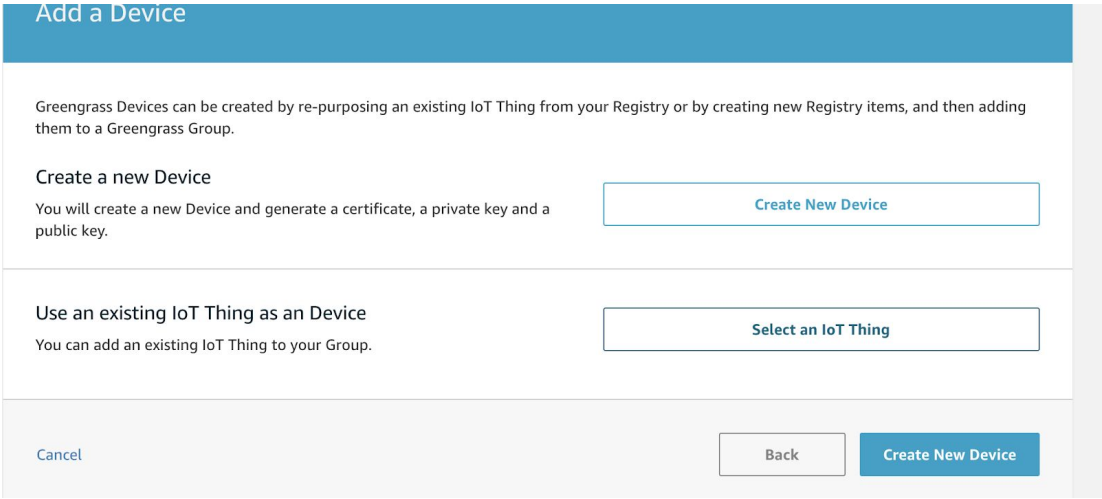
c). Use the laptop as an AWS IoT device.

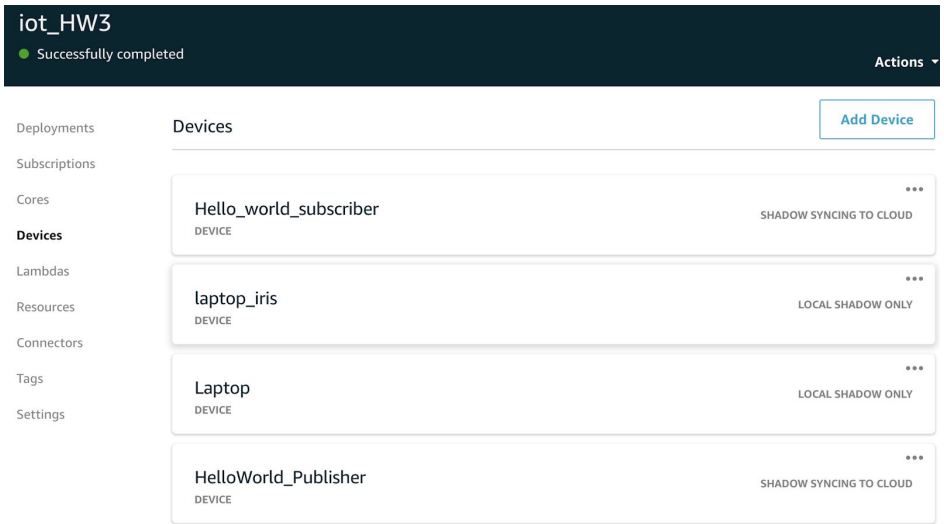
Register the devices in the Greengrass group as the thing and install AWS IoT SDK on the laptop and install AWS IoT SDK on this device.

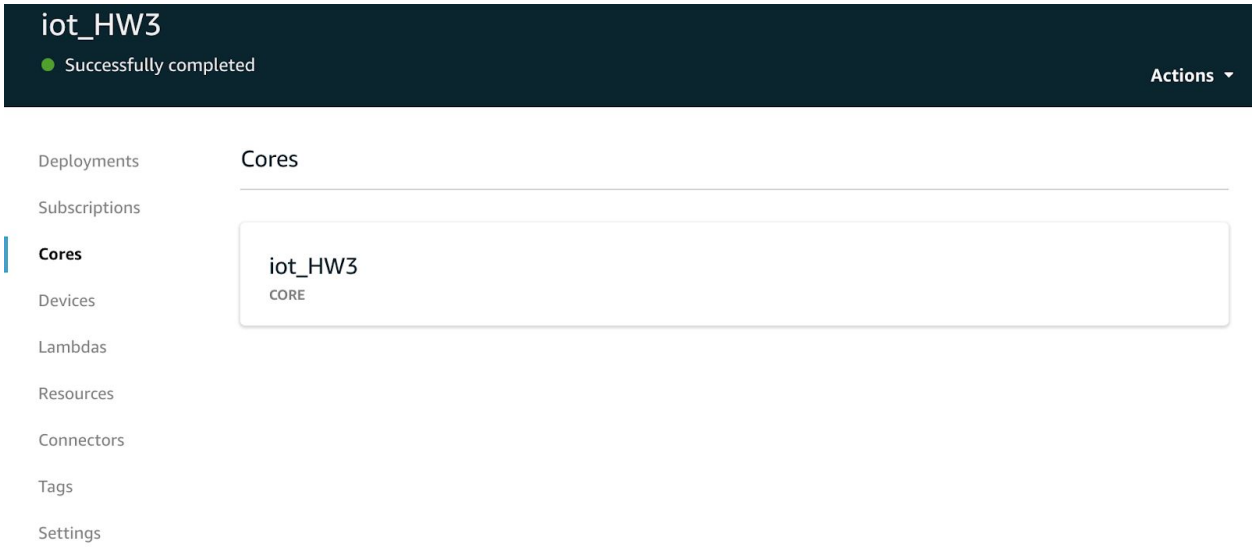


d). Add the core and IoT devices to the Greengrass group. Notice that you need to register your core and IoT device at first, then configure them (in AWS console) as core and IoT devices accordingly.

Register and add the core and devices to the IoT the Greengrass group.

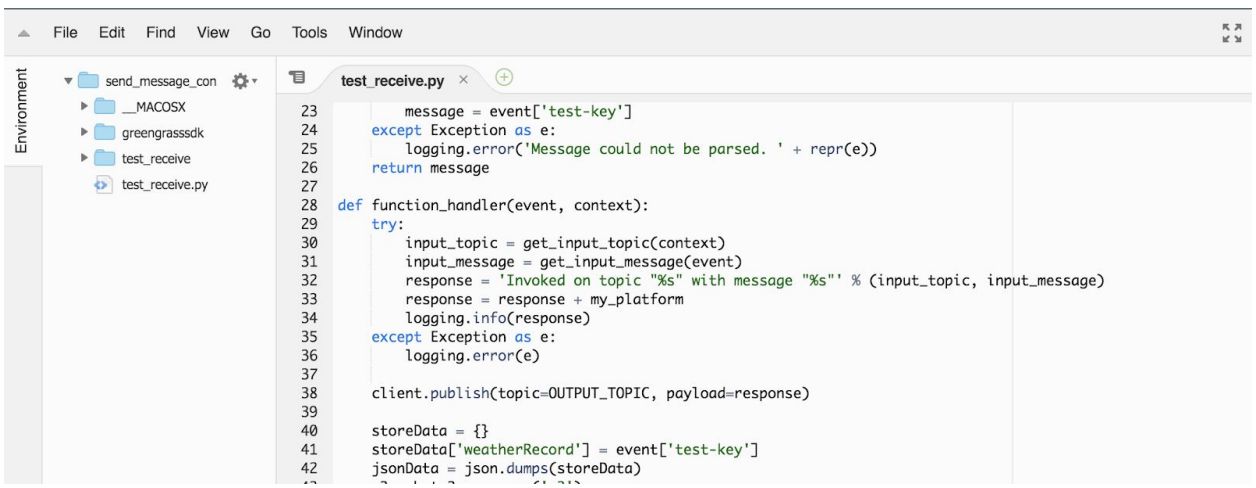


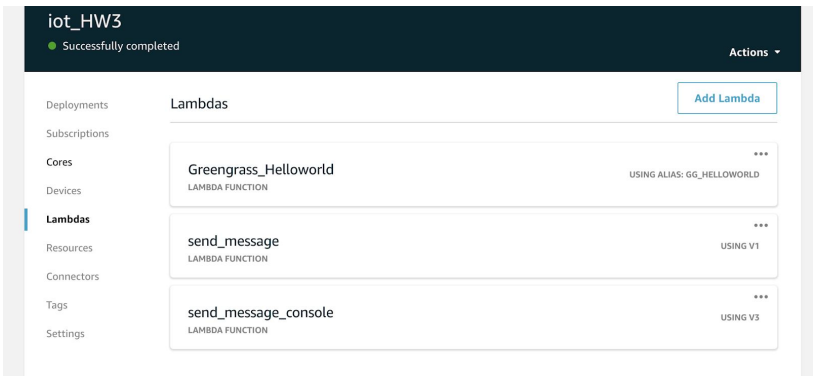


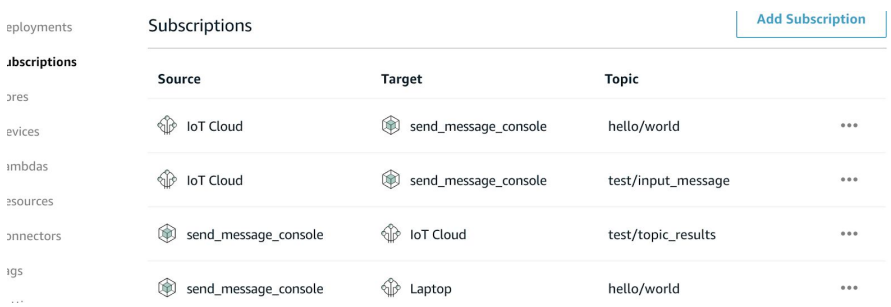


e). Develop and deploy a Lambda function named "LambdaCore" to the Greengrass Core

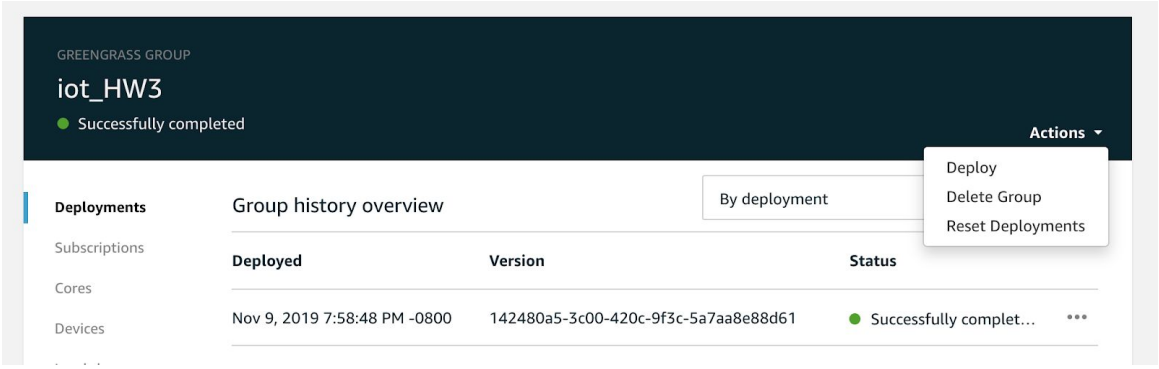
Create a Lambda function, zip python file, and Greengrass SDK and upload it to the lambda function. Set the lambda function in the Greengrass group page





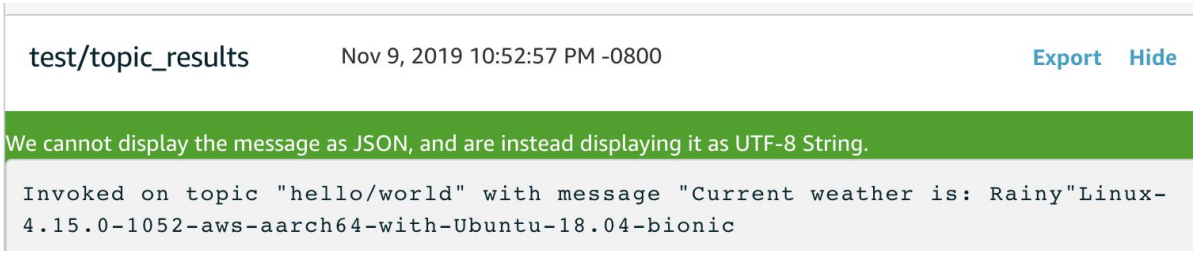


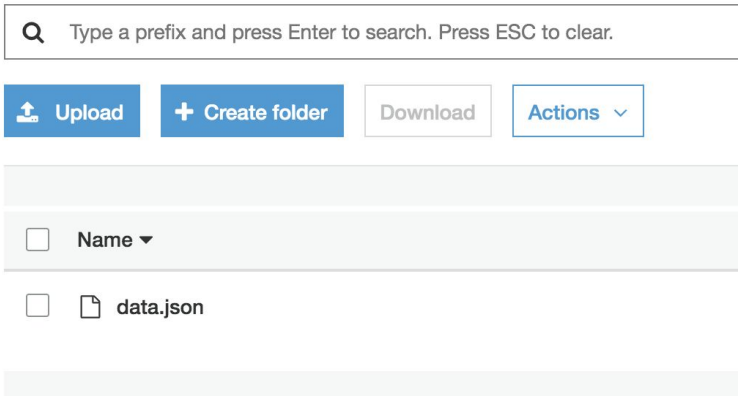
Creat subscription between devices(laptop), lambda function and the IoT console.



Run IoT Greengrass Daemon on the ec2 instance and deploy it.

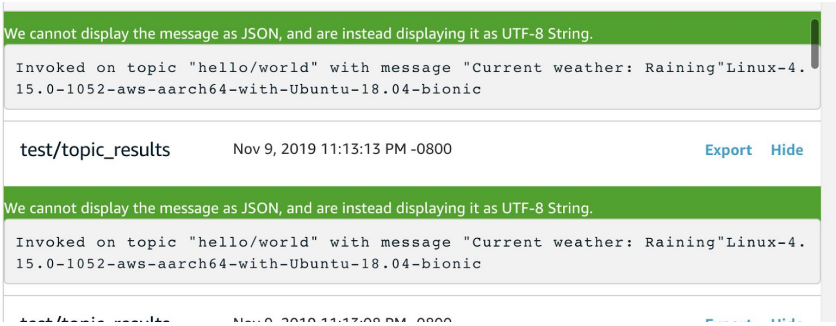
Run the python file sending the Message: Current weather and the console can receive the data. Data is stored into the s3 via the lambda function.





e) Develop a program running on the IoT device, the code is shown in the folder.

```
2019-11-09 23:06:03,629 - AWSIoTPythonSDK.core.protocol.mqtt_core - INFO - Performing sync publish...
2019-11-09 23:06:03,629 - AWSIoTPythonSDK.core.protocol.mqtt_core - INFO - Offline request detected!
2019-11-09 23:06:03,629 - AWSIoTPythonSDK.core.protocol.internal.queues - DEBUG - append: Add new element: <AWSIoTPythonSDK.core.pro
tocol.internal.requests.QueueableRequest object at 0x10576fcc0>
Published topic hello/world: {"test-key": "Current weather: Raining", "sequence": 18}
```



2 Problems Solving：

a). Install IoT Greengrass software package in ec2 instance

It took me a lot of time to configure the instance and install the correct version of software according to the system chosen for the ec2 instance. Also, authentication failed at the beginning, since the Root CA I used is AmazonRootCA3.pem at the beginning, but Greengrass only works well with AmazonRootCA1.pem.

b). Link to the core

I am a kind of confused since I am not able to link to the core via private IP address. The linking problem is fixed after I set up the connection manually and use the public IP address, but I am still trying to understand why I can only use public IP but most of the examples I see use private IP.

c). Deploy Failure

Deploy failed every time I update the lambda function version. The problem is fixed by deleting and recreating subscription. But I am still trying to find a more “reasonable” way to updating the lambda function without impacting the subscribing.

3 Time Consuming

Around 12 hours